PERSONATA

CS3216 Assignment 1 Pitch

Personata is an app that was created by us to help the user to evaluate the user's usage of Facebook. This application helps the user to know about the user's monthly posts, likes and comments as well as understand the time of the day when the user is the most active. Apart from this, the user will be able to share and invite their Facebook friends while also being able to view the usage statistics of their friends using the app.

User Interface:

Pages of the application have a really cool user interface due to implementations of CSS3. The main four layouts will adjust their sizes and visibility corresponding to the window size, i.e. the user interface is responsive. Moreover, four application buttons at main page are all neat and consistent, and when the user puts his/her mouse on the button, there will be an explanation of the functionality.

In addition to that, the loading indicator makes page transition smoother and provides better user experience. Graphs are intuitive and vivid, users can share these graphs to their feeds as well. Most popular posts are displayed by timeline style, which makes the result intuitive and pretty.

Performance:

We take the advantages of both Javascript and PHP SDK in our application. For posts data, which will not change with time, meaning number of posts for certain past dates cannot change, so we use PHP to retrieve and store it into the database. For likes and comments data, which change with time because people can like or comment on historical status etc, we use Javascript SDK to retrieve and process the data every time.

As for the storage in database, each user has one field called 'last update time' meaning this user's data is updated to that time. So next time when this user logs in, the new feed data between the last updated time and the current time is retrieved. This way we can reduce the loading time significantly.

When retrieving the number of likes and comments for a post, we limit the number of profile ids returned by Facebook to have liked the post to 1. In case this limit isn't specified, the returned object contains the names of all those who have liked the user's posts. This data is unnecessary for us and limiting it reduces the amount of data to be processed consequently saving time

We also cache the data for Javascript SDK to improve performance. Every time some data is fetched, it will be stored locally and when user browses that page again, graphs will be loaded from cached data directly and immediately for that page.

Answering the Aspirations 16 – 18 towards the coolness factor:

Aspiration 16:

Our application has a set of animations implemented on the application homepage. These animations were implemented using CSS3 and caused the icons to move when the user hovered close to the links. The piece of code for these set of animations is included below –

```
    -moz-animation: bg 500s linear infinite;
    -webkit-animation: bg 500s linear infinite;
    -o-animation: bg 500s linear infinite;
    -ms-animation: bg 500s linear infinite;
    animation: bg 500s linear infinite;
```

Aspiration 17:

Since we are using PHP SDK to retrieve and process data at server side, we use a lot of AJAX calls to fetch the data dynamically without reloading the page. For all the data are stored in database, we use the AJAX call to fetch the those data. The following are some examples:

```
$.getJSON( "backend.php", {data: 'post', uid: uid}, function( data ) {
}
```

Aspiration 18:

We have used jQuery plugin bootstrap and highcharts for responsive design and visualizing all the data we process in the form of user readable graphs.

```
<script src="js/bootstrap.min.js"></script>
<script src="js/highcharts.js"></script>
<script src="js/highcharts-more.js"></script>
<script src="js/exporting.js"></script>
<script src="js/heatmap.js"></script>
```